UCRL- 92360
PREPRINT

FAST MOVING AVERAGE RECURSIVE LEAST MEAN SQUARE FIT

Lawrence C. Ng

Robert A. LaTourette

This paper was prepared for submittal to
IEEE Conference on Decision and Control
Fort Lauderdale, Florida
December 11-13, 1985

March 13, 1985

FAST MOVING AVERAGE RECURSIVE LEAST MEAN SQUARE FIT [*]

(24th CDC Only)

Lawrence C. Ng [†]

Lawrence Livermore National Laboratory
University of California
P. O. Box 808, L-228
Livermore, California 94550

Robert A. LaTourette

Code 3213
Naval Underwater Systems Center
New London, Connecticut 06320

Abstract

A new approach is developed to reduce the computational complexity of a moving average Least Mean Square Fit (LMSF) procedure. For a long data window, a traditional batch approach would result in a large number of multiplication and add operations (*i.e.*, an order N, where N is the window length). This study shows that the moving average batch LMSF procedure could be made equivalent to a recursive process with identical filter memory length but at an order of reduction in computational load. The increase in speed due to reduced computation make the moving average LMSF procedure competitive for many real time processing applications. Finally, this paper will also address the numerical accuracy and stability of the algorithm.

Introduction

In processing a long stream of digital data, a polynomial moving average Least Mean Square Fit (LMSF) is often used to provide smoothing or filtering of the noisy component imbedded in the data [1]. The LMSF has a finite duration memory which is determined by the length of the moving window.

There are several advantages in using a moving average polynomial LMSF [2]. First and foremost is that LMSF provides a stable operation since the polynomial coefficients are

[*] Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

[†] Formerly with the Naval Underwater Systems Center, New London, CT 06320

obtained from a bank of Finite Impulse Response (FIR) filters operating on the data sequence. Secondly, the finite memory window assures us that bad data points lie outside the window will have no effect on the resulting LMSF estimates.

On the other hand, there are also a number of disadvantages in the moving average LMSF application. First is that the sizable amount of memory storage required for data lie within the operating window. Second is the apparent large computational requirement in comparison to the recursive implementation using Infinite Impulse Response filters [3]. The study presented here shows that while the size of storage requirement is unchanged, the computational load can be reduced significantly *via* an equivalent recursive formulation.

Methodology

Let $\mathbf{Z}_n$ denote a vector containing N consecutive data points from a measurement sequence, $\mathbf{A}_n$ denote a vector containing the coefficients from an M^{th} order polynomial, and H be the $N \times (M+1)$ dimensional matrix that relates $\mathbf{A}_n$ to $\mathbf{Z}_n$. Then the LMSF solution of $\mathbf{A}_n$ is given by:

$$\hat{\mathbf{A}}_n = \left(H^T H\right)^{-1} H^T \mathbf{Z}_n \quad . \tag{1}$$

This can also be written as

$$\hat{\mathbf{A}}_n = \left(H^T H\right)^{-1} \mathbf{X}_n \quad . \tag{2}$$

where

$$\mathbf{X}_n = H^T \mathbf{Z}_n \quad . \tag{3}$$

Note the subtle difference between Eqs. (1) and (2). First, $\mathbf{Z}_n$ is an N dimensional vector. For all practical purposes, N, the number of data points, is much larger than M, the order of the LMSF. Second, the matrix $(H^T H)^{-1} H^T$ in Eq. (1) is $(M+1) \times N$ dimension, and the matrix $(H^T H)^{-1}$ in Eq. (2) is $(M+1) \times (M-1)$ dimension. Thus assuming that $\mathbf{Z}_n$ and $\mathbf{X}_n$ are known, then computing $\mathbf{A}_n$ *via* Eq. (2) will result in savings by a reduction factor of $(M-1)/N$ of the number of multiplications and adds. For example, given a typical value of $N = 100$ and $M = 4$, we have $(M+1)/N = .05$. This is indeed a significant reduction in computation. Of course, the factor $(M+1)/N$ is the ideal lower bound since additional computations are required to obtain $\mathbf{X}_n$ from $\mathbf{Z}_n$. If $\mathbf{X}_n$ was obtain from $\mathbf{Z}_n$ directly using Eq. (3), then no reduction in computation is gained. Thus, it is desired to obtain an efficient computation of $\mathbf{X}_n$ from $\mathbf{Z}_n$. Consequently, we will obtain an efficient algorithm to calculate $\mathbf{A}_n$ for each moving window of length N. This has been accomplished using a recursive formulation [4]. The derivations are somewhat lengthy and therefore will not be shown here.

Speed Comparison

Both the batch and the recursive approaches were implemented on a VAX-11/780 machine. Using a number of different length windows, both approaches were used to process over 10,000 data points with the resulting CPU times carefully recorded. One can define the ratio of the recursive CPU time to the batch CPU time as the speed reduction ratio or speed ratio (SR) for short. Figure 1 shows a typical SR result. Also shown in Figure 1 is the lower bound, which was theorectically calculated and is given by the formula [4]:

$$LB = \left(\frac{Na}{Tm} + \frac{Nm}{Ta} \right) / (M - 1) \; N \quad . \tag{4}$$

where Na, Nm is the number of additions and multiplications respectively for the recursive approach. Ta and Tm are the machine cycle time required for a single add and multiply respectively.

Close study of Figure 1 shows that the actual SRs fall off as a function of window length at a rate similar to the lower bound. The significant difference between the actual SR and the lower bound is due to programming overhead cost; *i.e.*, CPU time expended for non-arithmetic operations. With more efficient programming, this difference is expected to reduce. At any rate, Figure 1 shows that at a window length of 200 data points, the recursive CPU time is only 20 percent of the batch CPU time. The corresponding number for the lower bound, however, is only 4 percent. On the other hand, for a small data window, say $N < 10$, the differences between the two approaches become insignificant.

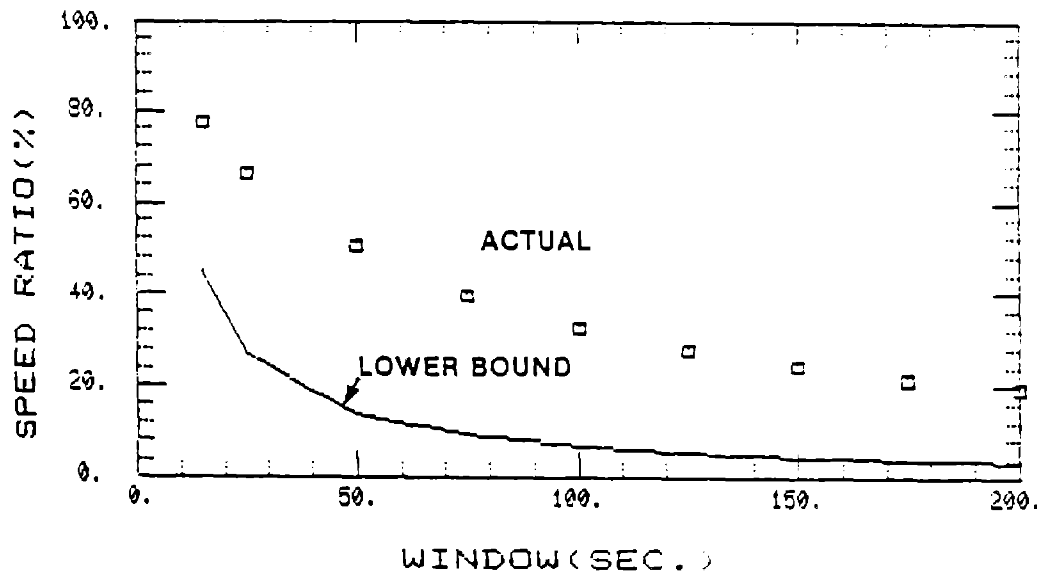

Figure 1. Speed reduction ratio versus window length

Summary and Conclusions

This study developed an efficient recursive algorithm to implement a moving average Least Mean Square Fit (LMSF) procedure. The following briefly summarizes the significant findings of this study.

1. A recursive formulation of a moving average LMSF can be implemented with a theorectical ratio in computation reduction (recursive over batch) of $(3M+4)/2N$, where M is the order of the LMSF and N is the window length.

2. Ignoring the potential singular value inversion and other numerical problems, recursive moving average LMSF gives outputs identical to the batch LMSF when the data window is filled. However, prior to attaining the full data window, the recursive approach has the additional advantage that it could also provide meaningful outputs if good a priori information is used to initiate the recursion.

3. We have shown the speed advantage of the recursive approach. However, we have not yet addressed the relative comparison from the numerical point of view. This important issue should be addressed. (study is currently in progress).

4. The significant reduction in computational load for the moving average LMSF makes it competitive for many real time processing applications.

References

1. L.C. Ng and R.A. LaTourette, "Equivalent Bandwidth of a General Class of Polynomial Smoothers," *J. Acoust. Soc. Am.* *74*(3), September 1983. Also NUSC Technical Report TR 6601, dtd. 19 July 1982.

2. N. Morrison, *Introduction to Sequential Smoothing and Prediction*, McGraw-Hill Book Company, 1969.

3. B.A. Bowen and W.R. Brown, *Signal Processing and Signal Processors*, Prentice-Hall, Inc., 1982.

4. L.C. Ng and P.R. Lambert, "Fast Moving Average Recursive Least Mean Square Fit," Naval Underwater Systems Center, Technical Memorandum TM 841143, dtd. 30 September 1984.